MODIS
Application Program Interface
User's Guide

Version 2.1

January 23, 1997

SDST-064A

MODIS
Application Program Interface
User's Guide, Version 2.1

Prepared By:

Frederick J. Shaw, SAIC/GSC Date
Software Programmer

MODIS
Application Program Interface
User's Guide, Version 2.0

Prepared By:

Nick Ruggiero, SAIC/GSC | Date
Senior Programmer

Reviewed By:

Barbara Putney, GSFC/Code 920 | Date
MODIS Systems Engineer

Laurie Schneider, SAIC/GSC | Date
SDST R&QA Manager

Frederick Patt, SAIC/GSC | Date
SDST Systems Analyst

Rich Isaacman, SAIC/GSC | Date
MODIS SDST Task Leader

Approved By:

Edward Masuoka, GSFC/Code 920.2 | Date
MODIS SDST Manager

Change Record Page

This document is baselined and has been placed under Configuration Control. Any changes to this document will need the approval of the Configuration Control Board.

Document Title: MODIS Application Program Interface User's Guide, Version 2.0

Document Date: August 21, 1996

Issue	Date	Page Affected	Description
Original	8/21/96	All	Baseline
Change Notice 1	10/10/96	6-4 thru 6-6c, 6-10 thru 6-13, C1, C6, C7, C14, C15	CCR #229
Revision A	1/23/97	All	CCR #274

MODIS
Application Program Interface
User's Guide, Version 2.1

Table of Contents

List of Figures

List of Tables

MODIS Application Program Interface User’s Guide, Version 2.1

1. INTRODUCTION

The Moderate Resolution Imaging Spectroradiometer (MODIS) Instrument Science Data Support Team (SDST) has the responsibility to integrate into a production environment atmosphere, land, and ocean remote sensing software provided by MODIS Science Team Members (STMs). The software is based on algorithms developed by the MODIS STMs and documented in the Algorithm Theoretical Basis Document (ATBD). General Sciences Corporation (GSC), a wholly owned subsidiary of Science Applications International Corporation (SAIC), is under contract with the National Aeronautics and Space Administration (NASA)/Goddard Space Flight Center (GSFC) to carry out the SDST functions.

The production environment for the integrated science software is the Distributed Active Archive Centers (DAACs), which are part of the Earth Observing System (EOS) Data and Information System (EOSDIS). The EOSDIS is developing recommendations regarding structures for the storage of scientific data in the Hierarchical Data Format (HDF), a self-describing format developed by the National Center for Supercomputing Applications (NCSA) (Getting Started with HDF, Draft, Version 3.2, May, 1993). HDF was chosen as the storage structure by EOSDIS after a nearly four-year effort surveying input from several of the DAACs, EOS instrument investigators, other Earth science projects, scientists outside of the United States, computer scientists, and other members of the EOS community. HDF structures are already in use or being planned for use on the Advanced Very High Resolution Radiometer (AVHRR)/Pathfinder, the Tropical Rainfall Measuring Mission (TRMM), and the Sea-viewing Wide Field-of-view Sensor (SeaWiFS) Projects.

1.1 The Purpose of MODIS-Application Program Interface

This document describes the Version 2.1 Release of the MODIS-Applications Programming Interface (M-API), developed by the MODIS SDST, for both reading and writing of science data sets associated with land, ocean, and atmosphere algorithms. The intended audience includes both scientists and programmers building algorithm-based software as part of the MODIS Version 1 (V1) code delivery to the EOSDIS Core System (ECS). A knowledge of C or FORTRAN programming and a general knowledge of HDF would be helpful but is not required to start using M-API.

The M-API is designed to greatly simplify the process of reading Level 1B (L1B) radiance bands and Level 2 (L2) science data; and for writing output products and metadata to the HDF files. The metadata consists of both a core set and a product-specific set, using the standards enumerated by the ECS group. The M-API shields the users as much as possible from the low-level details of HDF, so they can focus on their science. At the same time, the M-API allows a great deal of flexibility so that the users can customize the structure of the HDF output to meet their own research needs. The M-API handles the generation of low-level objects within the HDF and organizes them, as efficiently as possible, to access to their contents by other L2, as well as Level 3 (L3), algorithms.

In the original requirements document for M-API, performance was not listed as one of the major design factors. The previous versions of M-API do not consider M-API's performance. The recent user feedback showed significant performance deterioration compared to using HDF functions directly, therefore, M-API 2.1 has been designed to correct this performance deterioration.

Figure 1-1 shows the relationship between user programs, M-API, SDP Toolkit, and HDF.

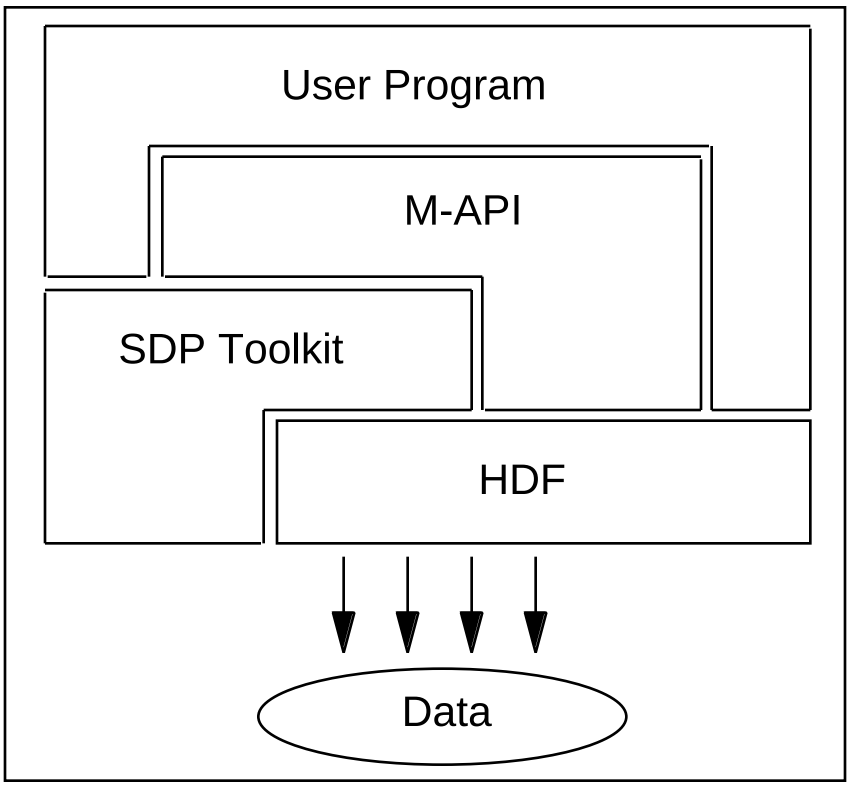

Figure 1-1. Applications

1.2 Organization of the M-API User’s Guide

The M-API User’s Guide is organized as follows:

- Section 1 is an overview of M-API and its purpose.
- Section 2 lists related documents for the user’s reference.
- Section 3 describes specific M-API capabilities, Version 2.1 restrictions, the procedures for installing the M-API software and compiling and linking applications programs which call the M-API.
- Section 4 provides instructions on using M-API to access data arrays, access data tables, and using metadata.
- Section 5 describes the M-API constants and naming conventions, and examples of MODIS file structures.
- Appendix A is the listing of acronyms.
- Appendix B provides the M-API Supplied Constants.
- Appendix C contains the M-API module descriptions.
- Appendix D provides the variables for each of the M-API routines, both for C and FORTRAN language interfaces.
- Appendix E provides error messages associated with each of the M-API routines, both for C and FORTRAN language interfaces.
- Appendix F provides examples of the MODIS Product File Definitions.

(This page intentionally left blank)

2. RELATED DOCUMENTS

2.1 Hierarchical Data Format-Related Documents

- Getting Started with HDF, Draft - An overview of HDF and how to create programs for reading and writing HDF files. We recommend a reading of this document to M-API users new to HDF.
- HDF Reference Manual - Detailed descriptions of the HDF C and FORTRAN subroutines.
- HDF-EOS Draft Standard - The ECS view of what capabilities they will support, including descriptions of an HDF taxonomy including swaths, grids, point data, and metadata.

2.2 EOSDIS Core System-Related Documents

- SDP Toolkit User's Guide for the ECS Project - Descriptions of routines which are part of the EOS Core System's Science Data Processing (SDP) Toolkit. The SDP Toolkit will, in a future release, have routines for retrieving metadata values from HDF files and writing them to the files.
- ECS Science Software Delivery Procedures - The ECS view of how science software is to be delivered to the DAAC for integration into a production system.
- ECS Metadata Syntax: Granules - A description of the metadata ECS expects to see at the granule (i.e., HDF file) level. This can be used as a reference in formulating the metadata to include in an HDF file as global attributes.
- Global Change Master Directory (GCMD) Data Interchange Format (DIF) manual - This is referenced by the ECS Metadata Syntax: Granules document as defining the valid values for each of the metadata items.
- PVL-ODL User's Guide - Descriptions of the Parameter Value Language (PVL) and Object Description Language (ODL), which are the basis for the format of the metadata items to be included in the HDF files.

2.3 Other Documentation

- MODIS Standards and Guidelines - The document describing SDST's expectations for how delivered science code will look.
- World Wide Web (WWW) Uniform Resource Locators (URLs):
  - http://ltpwww.gsfc.nasa.gov - Laboratory for Terrestrial Physics home page.
  - http://newsroom.hitc.com/sdptoolkit/toolkit.html - SDP Toolkit home page.

- http://edhs1.gsfc.nasa.gov - ECS Data Handling System (on-line repository of ECS documents).
- http://ltpwww.gsfc.nasa.gov/MODIS/MODIS.html - MODIS home page.
- http://ltpwww.gsfc.nasa.gov/MODIS/SDST/Home.html - SDST home page.
- http://ltpwww.gsfc.nasa.gov/IS/SDST/stig.html - SDST Science Software Transfer Group (SSTG) home page.

3. M-API FEATURES AND INSTALLATION

3.1 M-API Features

M-API will offer the following features when it is fully implemented:

- Hardware Support - All of the platforms currently employed at the MODIS Science Computing Facilities (SCFs), including Silicon Graphics Incorporated (SGI), Sun, Hewlett Packard, IBM RS/6000 and the Digital Equipment Corporation (DEC)/Alpha. A customized make file will be supplied for installing the software on each of these platforms.
- Language Support - There will be separate libraries for C, FORTRAN77, and FORTRAN90. FORTRAN algorithms submitted for the V1 software release need not use FORTRAN90, although it is now supported as a standard by Earth Science Data and Information System (ESDIS) (MODIS Standards and Guidelines, 1995).
- Accessible Data - Any portion of an array, subsetted along any dimension, or any portion of a table, subsetted by a record or a table entry. Data can be extracted from or written to arrays or tables. Data representing multiple storage types, including float, double precision, short or long integers, and bytes can be stored within a table or across multiple arrays/objects in the HDF file.
- Metadata Support - Reading and writing of both ECS core-level and product-specific attributes at the HDF file and the object level.
- Low-Level Functions - Open and close files, initialize array structures, create Vgroup structures, create tables, query array information, query table information, read and write metadata, read and write tables, and read and write arrays stored in HDF Scientific Data Sets (SDS).

For M-API Version 2.1, the following new routines have been added:

- endMODISobjaccess (EMOBJ),
- getMODISdimname (GMDNAM), and
- putMODISdimname (PMDNAM).

3.2 M-API and Hierarchical Data Format

The M-API utilities are what is described in HDF documentation as a *3rd Party Application*. (Getting Started with HDF, Draft, Version 3.2). It uses the standard HDF interface to store MODIS data in standard HDF formats. Multidimensional arrays are accessed using HDF's Scientific Data Set (SD) interface for SDS. The SD interface is also used to access file metadata in global attribute records. Tables of scientific data are stored as Vdata using the Vdata Set (VS) interface. Various data objects may be collected into a data group and identified as members of a Vgroup by the Vgroup (V) interface. Since M-API utilities use standard HDF structures to store data, MODIS Data Products are accessible to other NCSA HDF applications, commercial packages that support HDF, and ECS utilities.

One important characteristic of M-API utilities is the ease of use it provides as an Input/Output (I/O) interface. The HDF application interfaces use a family of reference identifiers ('refids') and tags. These are 32-bit integers to identify objects, their types, and to refer to those objects that are currently 'open' for access. M-API utilities use American Standard for Computer Information Interchange (ASCII) strings to identify MODIS data objects. The utilities handle the 'bookkeeping' of refids, HDF file searching, and data access identifiers. They are also 'complete', which is to say that any data object that is written to an HDF file using the M-API utilities can also be read by them. The M-API utilities provide all of the functions required to access MODIS data products. So even though the M-API provides for portable MODIS data products it is not necessary for programmers to become familiar with HDF, beyond installing the HDF library on their development computer, compiling the M-API utilities with the HDF headers 'included', and linking the HDF libraries with their MODIS product software.

3.3 Restrictions

The M-API utilities are not, and are not meant to be, a general HDF application. M-API was designed to support the access to MODIS data products by the MODIS algorithm processes. It uses a limited set of HDF ‘data models’ and features that are sufficient to support the needs of MODIS science data processing. It does not use several HDF data models, such as raster, raster palettes, or multiple file SDS’s. Any of these HDF facilities may be incorporated into the M-API utilities in a later version if is determined that they are required.

The M-API utilities use ASCII string names to reference the various data objects in MODIS product files. The question “Where is it located?”, required to find data in a traditional L2 file, is replaced in the M-API paradigm by “What is it called?” Unfortunately, name strings may be easy for humans to use, but are a challenge for computers to handle. Mismatches due to differences in case, hyphens, underscores, and even blanks, can easily occur. This is one reason M-API supplies a standard set of constant string names (‘macros’, ‘symbolic constants’) to be used for accessing MODIS data objects. These constants are contained in the ‘mapi.h’ header file that must be included in each MODIS algorithm routine calling M-API utilities. The name strings include MODIS product names, data object names, table headers, array labels, and metadata names. It is an important facet of the M-API that these constants be used to access data objects, so that consistent naming conventions (product specific include files) may be maintained and data may be retrieved reliably.

Because the M-API uses string names to identify data objects, difficulties may arise if two of the same type of objects have the same name in a MODIS product file. The M-API utilities will normally prevent this name contention. It is permitted to have like-named objects in a file if they are placed in separate data groups (Vgroups), however.

3.4 Installing M-API

The M-API Version 2.1 is available from the MODIS SDST as a single UNIX tar file. The M-API, along with the required NCSA HDF software, are retrieved through File Transfer Protocol (FTP) commands from any UNIX machine. Table 3-2 shows the compatibility of the hardware and software.

Table 3-2. Hardware/Software Compatibility

Hardware	Operating Software	C Compiler	FORTRAN 77	HDF	SDP TK	Comments
DEC	OSF1 3.2	cc	F77	4.0r1P1	5.0/5.1	OK
HP						
IBM	AIX 2	cc	XLF	4.0r1P1	5.0/5.1	OK
SGI	-32 IRIX64 6.2	cc	F77	4.0r1P1	5.0/5.1	OK
	-n32 IRIX64 6.2	cc	F77	4.0r1P1	5.0/5.1	OK
	-64 IRIX64 6.2	cc	F77	4.0r1P1	5.0/5.1	OK
SUN	SunOS 5.4	cc	F77	4.0r1P1	5.0/5.1	OK

NOTE: FORTRAN 90 compatibility has not been tested.

To successfully download and uncompress the M-API, a minimum of 250 kilobytes of free space is required. The Version 2.1 M-API release has been developed and tested in American National Standards Institute (ANSI) C on SGI workstations running the IRIX Version 6.2 operating system.

3.4.1 System Requirements

The following are system requirements that must be met in order to install the complete M-API:

- The system must have ANSI compliant C and FORTRAN77 compilers.
- The system must have about 75 MB of free disk space (primarily for installing HDF).
- HDF 4.0r1P1 or HDF 3.3r4 should already be installed.
- Science Data Processing Segment (SDPS) Toolkit 5.0 or 5.1 should already be installed.

3.4.2 Installing the M-API Software Library

The M-API software distribution is available via anonymous FTP from the Team Leader Computing Facility (TLCF) at NASA/GSFC at:

1. Internet Protocol (IP) Address: ltpftp.gsfc.nasa.gov (128.183.76.153)

2. Directory: /pub/projects/modis/util/modis_api

To install M-API on your local machine:

· Download the M-API software distribution from NASA/GSFC, using FTP or analogous software.

  For example:

```
ftp ltpftp.gsfc.nasa.gov
```

  At the "Name:" prompt, enter:

```
anonymous
```

  At the "Password:" prompt, enter your full e-mail address.

  Change directory:

```
cd /pub/projects/modis/util/modis_api
```

· Next, download the tar file containing the M-API version you wish to install. For example:

```
get mapi2.0.tar.Z

get mapi2.UG.pdf.Z or the WORD version: get mapi2.UG.doc.hqx
```

  When the downloading is completed, exit ftp.

```
quit
```

· Uncompress and unpack the M-API software distribution.

First, select a directory that will contain the M-API software. Move the downloaded tar file to this directory, then uncompress and unpack the tar file.

```
uncompress mapi2.0tar.Z
tar xvf mapi2.0.tar
```

The tar command will create a sub-directory called "mapi2.0" which contains the entire M-API software distribution. Inside the "mapi2.0" directory are the following files and directories:

1. README : General Information
2. Makefile.sgi : Makefile used to build the M-API on an SGI
3. examples/ : Directory containing several example programs
4. h/ : Directory containing M-API include files
5. srclib/ : Directory containing M-API source
6. lib/ : Directory containing M-API Library

NOTE: The SGI Power Challenge at the TLCF uses 32bit, n32bit, 64bit to denote the library directory appropriate to that Application Binary Interface (ABI).

- Edit the appropriate makefile to your platform, noting your local directories of HDF PGS Toolkit, and M-API.
- Then, use Make to compile the M-API archive library, libmapi.a:

```
make -f makemapi.sgi all
```

After successfully compiling the M-API archive library, the installation is complete.

3.5 M-API in User Programs

The 'examples' sub-directory of the M-API distribution contains brief demonstration programs that exercise some of the more essential M-API utilities. The Makefile that compiles and links these programs also demonstrates the necessary compiler inclusions required for creating executable M-API programs. A more complete description can be found in Section 9.

The M-API utilities consist of a set of routines designed to be called directly from the scientific algorithm software. There are routines for calling from both the C and FORTRAN77 language. They are to be used much like standard I/O routines to open, write, read, and close data files. Section 5 provides an alphabetical reference, including the calling syntax, of the M-API utilities.

Any routine that uses a M-API utility, constant, or data type must include the M-API header file that contains information needed to use these objects. There is a mapi.h file to include in such C routines and an analogous mapi.inc file for FORTRAN routines. These headers do not have nested includes to the HDF header files, so any explicit use of HDF utilities, constants or data types will require HDF headers to be included as well. For portability, the include statement should reference the name of the header file only, and not the path to it:

```
#include "mapi.h"
```

The path to the M-API header file must be supplied to the compiler when compiling a routine containing the M-API header file:

```
cc -ansi -I $(API_INC) -c myprog.c
```

The 'API_INC' environment variable contains the directory path of the M-API header files and must be defined prior to using it in the compilation. Unless a routine uses HDF facilities explicitly, there is no need to provide the directory path to the HDF header files.

The program must be linked to both the M-API and HDF object libraries when making an executable using HDF 4.0:

```
cc myprog.o -L$(API_LIB) -lmapi -L$(HDFLIB) -lmfhdf -ldf -1jpeg
-lz -L$(PGSLIB) -lPGSTK -lm -o myprog
```

The three environment variables 'API_LIB', 'PGSLIB', and 'HDFLIB' are defined by the user. Additional libraries may be included in the link as well. The ordering of the libraries should be retained by the developer in order to avert unresolved references.

Below is a sample of the UNIX environment variables that are typically used at the MODIS TLCF (note the setup reflects a choice of -n32 on the SGI Power Challenge).

```
PGSBIN=/cm/tools/src/SDPTK5.1v1.01/bin/sgi32
PGSDAT=/cm/tools/src/SDPTK5.1v1.01/database/sgi32
PGSINC=/cm/tools/src/SDPTK5.1v1.01/include
PGSLIB=/cm/tools/src/SDPTK5.1v1.01/lib/sgi32
PGSMSG=/cm/tools/src/SDPTK5.1v1.01/message
PGSOBJ=/cm/tools/src/SDPTK5.1v1.01/obj/sgi32
PGSRUN=/cm/tools/src/SDPTK5.1v1.01/runtime
PGSSRC=/cm/tools/src/SDPTK5.1v1.01/src
PGSTST=/cm/tools/src/SDPTK5.1v1.01/test
HDFHOME=/cm/tools/libs/n32bit/HDF4.0r1p1
HDFBIN=/cm/tools/libs/n32bit/HDF4.0r1p1/hdf/bin
HDFINC=/cm/tools/libs/n32bit/HDF4.0r1p1/hdf/include
HDFLIB=/cm/tools/libs/n32bit/HDF4.0r1p1/hdf/lib
API_INC=/cm/tools/src/MAPI2.1/h
API_LIB=/cm/tools/src/MAPI2.1/n32bit
```

NOTE: Depending on your application, other PGS toolkit environment variables may be needed at run-time.